Theoretical Computer Science 125 (1994) 373–383
Elsevier

Note

The communication complexity of computing differentiable functions in a multicomputer network

Pengyuan Chen

Department of Mathematics and Computer Science, Gustavus Adolphus College, Saint Peter, MN 56082, USA

Communicated by M. Nivat
Received June 1992

Abstract

Chen, P., The communication complexity of computing differentiable functions in a multicomputer network, Theoretical Computer Science 125 (1994) 373–383.

Employing Abelson's multistage model of distributed computation, we study the amount of intercomputer communication required for a differentiable function to be computed in a multicomputer network, under the assumption that each computer in the network has only partial information about the input data. We formulate a general approach for obtaining a lower bound on the amount of required information exchange among the computers in the network. Unlike other studies in this setting, where the lower bound results are obtained only for the special two-computer case and often for a certain particular class of functions, our approach makes it possible to obtain a lower bound result for a general multicomputer network and for any differentiable function. The lower bound obtained here is given in terms of a certain differential-geometric property of the function to be computed and is not merely asymptotic, in contrast to most lower bound results in the literature. Furthermore, the lower bound is fully constructive and computable.

1. Introduction

Many computation problems can be formulated as problems of computing a function. In many practical situations, the input data for such a computation are initially

Correspondence to: P. Chen, Department of Mathematics and Computer Science, Gustavus Adolphus College, Saint Peter, MN 56082, USA. Email: chen@nic.gac.edu.

SSDI 0304-3975(93)E0152-T

dispersed over a computer network. In such a situation, the desired computation can be carried out distributively by the computers in the network with the necessary communication among them. It is well realized that the intercomputer communication required for such a computation can be a prominent factor in the computation time for certain problems (see e.g. [5]). In the light of this fact, distributed algorithms for this type of problems should be designed in such a fashion that the intercomputer communication is kept to a minimum. To this end, a good understanding of the communication complexity, i.e. the minimum communication required, of the computation problem is needed.

In this paper we pursue this line of inquiry for the general problem of computing a differentiable function under the assumption that the input data are dispersed among the computers in the network and aim at determining the minimum communication required for such a computation. Unlike most studies on communication complexity in the literature, our work is in a continuous and differentiable setting. The model we use is due to Abelson [1], whose main features are the following. The computation by and communication between the computers are modelled as a synchronized multistage process; the messages are assumed to be real valued and the communication protocols are assumed to be differentiable functions. In this setting, we formulate a general approach for obtaining a lower bound on the amount of information exchange among the computers required for the network to perform the desired computation. Unlike other studies along this line (e.g. [1, 6, 7]), where lower bound results are obtained only for the special two-computer case and often for a certain particular class of functions, our approach makes it possible to obtain a lower bound result for a general multicomputer network and for any differentiable function. The lower bound obtained here is given in terms of a certain differential-geometric property of the function to be computed and is not merely asymptotic, in contrast to most lower bound results in the literature. Another feature of the lower bound is that it is universal in the sense that it does not depend on the interconnection configuration of the network nor on which computers are to output what results. Furthermore, our lower bound is fully constructive and computable.

The paper is organized as follows. Abelson's multistage model of distributed computation is recalled in Section 2. Our approach is presented and the lower bound is obtained in Section 3. In Section 4, the computability of the lower bound is demonstrated and applications to several matrix computation problems are illustrated. Finally in Section 5, some concluding remarks are offered.

2. Abelson's multistage model of distributed computation

Suppose we want to compute a function $\boldsymbol{F}: \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_s} \to \mathbb{R}^m$ in a network of s fully interconnected computers, denoted by $\mathrm{CX}_1, \ldots, \mathrm{CX}_s$, and suppose that computer CX_i has access to data $\boldsymbol{x}_i \in \mathbb{R}^{k_i}$ only, $i = 1, \ldots, s$. The distributed computation of $\boldsymbol{F}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s)$ is modelled by a synchronized multistage process. Each stage consists of

a computation phase and a communication phase. Every computer in its computation phase performs some computation based on its private data and messages received from the other computers in the previous stages; every computer in its communication phase communicates with the other computers by sending and receiving messages. During each stage all computers compute and then communicate concurrently; the whole process is synchronized by the stages. At the initial stage, every computer computes some values solely based on its private data and then sends them to other computers. Let $f_{ij}^1(x_i)$ denote the (real) values that CX_i computes and sends to CX_j at the initial stage. At the second stage, every computer computes some (real) values based on its private data and the messages received from the other computers at the initial stage and then sends them to the other computers. Let $f_{ij}^2(x_i; m_{1i}^1, \ldots, m_{si}^1)$ denote the values that CX_i computes and sends to CX_j at the second stage, where $m_{li}^1 = f_{li}^1(x_l)$ denotes the messages that CX_i receives from CX_l at the initial stage. In general, at the kth stage, every computer computes some values based on its private data and the messages received from the other computers in the previous $k-1$ stages and then sends them to the other computers. Let $f_{ij}^k(x_i; m_{1i}^1, \ldots, m_{si}^1, \ldots, m_{1i}^{k-1}, \ldots, m_{si}^{k-1})$ denote the values that CX_i computes and sends to CX_j at the kth stage, where $m_{li}^p = f_{li}^p$ denotes the messages that CX_i receives from CX_l at the pth stage. This process is to continue until the values $F(x_1, \ldots, x_s)$ of the function F can be computed and output by some computer at some, say the rth, stage. (A minor extension can be made so that the values $F(x_1, \ldots, x_s)$ are allowed to be output by different computers at different stages. Our result still holds, as will be clear later.) The function f_{ij}^k are called *communication protocols.* Typically, some kind of regularity assumption should be imposed on communication protocols to avoid uninteresting cases. In this paper they are assumed to be differentiable functions. This implies that we assume in this idealized model all computers are able to compute the differentiable function and to communicate the real number to perfect precision. For a given computation problem F, what the communication protocols should be is a design issue for such a computation process, which can be viewed as a distributed algorithm for computing F. Without loss of generality, assume that the values $F(x_1, \ldots, x_s)$ are finally computed and output by CX_1. $F(x_1, \ldots, x_s) = f^r(x_1; m_{21}^1, \ldots, m_{s1}^1, \ldots, m_{21}^{r-1}, \ldots, m_{s1}^{r-1})$. Let a_{ij}^k denote the number of components in function f_{ij}^k. The *communication complexity* N_1 of this process is defined to be the total number of real-valued messages that need to be transmitted between the computers in the network in order to compute the values $F(x_1, \ldots, x_s)$, namely,

$$N_1 = \sum_{k=1}^{r-2} \sum_{i \neq j} a_{ij}^k + \sum_{i>1} a_{i1}^{r-1}.$$

The subscript 1 indicates that CX_1 is used to output the result. Note that the messages m_{ij}^{r-1} for $j \neq 1$ are not counted because they are not used in computing $F(x_1, \ldots, x_s)$, so there is no need to transmit them. The *communication complexity* of computing F in such a distributed network is defined to be the minimum of the communication complexity of all multistage processes that compute F.

To illustrate, consider computing in a three-computer network function $F: \mathbb{R}^{k_1} \times \mathbb{R}^{k_2} \times \mathbb{R}^{k_3} \to \mathbb{R}$, defined by

$$F(x,y,z) = \sum_{i=1}^{k_1} x_i (p_2(y_1))^i + \sum_{i=1}^{k_2} y_i (p_3(z_1))^i + \sum_{i=1}^{k_3} z_i (p_1(x_1))^i,$$

where p_1, p_2, p_3 are functions. Assume that the computers CX, CY, CZ have access only to the partial data x, y, z, respectively. This function can be computed by a three-stage process as follows. At the first stage, CX computes $p_1(x_1)$ and then sends it to CZ; CY computes $p_2(y_1)$ and then sends it to CX; CZ computes $p_3(z_1)$ and then sends it to CY. At the second stage, CX, CY, CZ are able to compute $\sum_{i=1}^{k_1} x_i (p_2(y_1))^i$, $\sum_{i=1}^{k_2} y_i (p_3(z_1))^i$, $\sum_{i=1}^{k_3} z_i (p_1(x_1))^i$, respectively, with their partial data and the messages received at stage one. Then by having, for example, CY and CZ send $\sum_{i=1}^{k_2} y_i (p_3(z_1))^i$ and $\sum_{i=1}^{k_3} z_i (p_1(x_1))^i$, respectively, to CX, CX is able to compute and output the result $F(x,y,z)$ at the third stage. We see that in this process 5 messages must be transmitted between CX, CY, and CZ, so the communication complexity of this process is 5. A question naturally arises. Is this the best that we can do in terms of the number of messages transmitted? Is there a multistage process that computes F with communication complexity less than 5? What is the communication complexity of computing this function F? These questions will be answered later by our lower bound result.

3. The equilibrium model and the lower bound

Suppose $\boldsymbol{F}: \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_s} \to \mathbb{R}^m$ can be computed by a multistage process with communication complexity $N_1 = \sum_{k=1}^{r-2} \sum_{i \neq j} a_{ij}^k + \sum_{i>1} a_{i1}^{r-1}$, as described in Section 2. Let $\boldsymbol{m}_{ij}^k$ denote the message variables and $\boldsymbol{m}^r$ the output; then we have

$$\boldsymbol{m}_{ij}^1 = \boldsymbol{f}_{ij}^1(\boldsymbol{x}_i), \quad i \neq j,$$

$$\boldsymbol{m}_{ij}^k = \boldsymbol{f}_{ij}^k(\boldsymbol{x}_i; \boldsymbol{m}_{1i}^1, \ldots, \boldsymbol{m}_{si}^1, \ldots, \boldsymbol{m}_{1i}^{k-1}, \ldots, \boldsymbol{m}_{si}^{k-1}), \quad i \neq j,\ k = 2, \ldots, r-2,$$

$$\boldsymbol{m}_{i1}^{r-1} = \boldsymbol{f}_{i1}^{r-1}(\boldsymbol{x}_i; \boldsymbol{m}_{1i}^1, \ldots, \boldsymbol{m}_{si}^1, \ldots, \boldsymbol{m}_{1i}^{r-2}, \ldots, \boldsymbol{m}_{si}^{r-2}), \quad i > 1,$$

$$\boldsymbol{m}^r = \boldsymbol{f}^r(\boldsymbol{x}_1; \boldsymbol{m}_{21}^1, \ldots, \boldsymbol{m}_{s1}^1, \ldots, \boldsymbol{m}_{21}^{r-1}, \ldots, \boldsymbol{m}_{s1}^{r-1}).$$

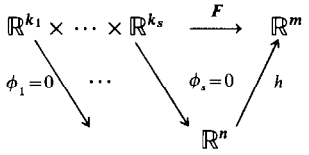
This set of equations can be rewritten as

$$\phi_i(\boldsymbol{x}_i; \mu) = 0, \quad i = 1, \ldots, s, \tag{3.1}$$

where $\mu = (\boldsymbol{m}_{ij}^k, \boldsymbol{m}^r)$, which has $N_1 + m$ components, is the vector of message variables, $\phi_1(\boldsymbol{x}_1, \mu) = (\boldsymbol{m}_{1j}^k - \boldsymbol{f}_{1j}^k, \boldsymbol{m}^r - \boldsymbol{f}^r)$, which has $\sum_{k=1}^{r-2} \sum_{j>1} a_{1j}^k + m$ components, is the collection of equations that contain $\boldsymbol{x}_1$ explicitly, and $\phi_i(\boldsymbol{x}_i, \mu) = (\boldsymbol{m}_{ij}^k - \boldsymbol{f}_{ij}^k)$, which has $\sum_{k=1}^{r-2} \sum_{j \neq i} a_{ij}^k + a_{i1}^{r-1}$ components, is the collection of equations that contain $\boldsymbol{x}_i$ explicitly, $i = 2, \ldots, s$.

For given data $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s)$, the message variables μ are uniquely determined in (3.1). The values of $\boldsymbol{F}$, $\boldsymbol{F}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s)$, can be computed simply from the messages μ, namely,

$\boldsymbol{F}(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_s)=\boldsymbol{m}^r=h(\mu)$, where h is an appropriate projection function. This relationship can be summarized succinctly in the following so-called Mount–Reiter [8] diagram, which has been used to study resource allocation processes in economics:



(3.2)

where $n=N_1+m$.

The Mount–Reiter diagram models the notion of integrating distributed information via messages and then converting these messages into the output. It is a commutative diagram in the following sense. For each set of data $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_s)$, $\boldsymbol{F}(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_s)$ can be computed by first solving for the message variables μ in (3.1) and then evaluating the function h on the messages μ, i.e. $\boldsymbol{F}(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_s)=h(\mu)$, where μ is solved from (3.1). For our present purpose, we call $(\phi_1,\ldots,\phi_s;\mathbb{R}^n;h)$ an *equilibrium model* of computing $\boldsymbol{F}$ if the commutative diagram (3.2) holds, and we call $\mathbb{R}^n$ its *message space.*

From the above description, we see that a multistage process of computing $\boldsymbol{F}$ leads naturally to an equilibrium model of computing $\boldsymbol{F}$. Note that the dimension n of the resulting message space is exactly the number of message variables, i.e. N_1+m. So we see that a multistage process of computing $\boldsymbol{F}$ with communication complexity N can be converted into an equilibrium model of computing $\boldsymbol{F}$ with a message space of dimension $N+m$. Hence, we have proved the following proposition.

Proposition 3.1. *Let* $\boldsymbol{F}\colon \mathbb{R}^{k_1}\times\cdots\times\mathbb{R}^{k_s}\longrightarrow\mathbb{R}^m$ *be a function. If* $\boldsymbol{F}$ *can be computed by a multistage process with communication complexity* N*, then* $\boldsymbol{F}$ *can be computed by an equilibrium model with a message space of dimension* $N+m$.

For what follows, we shall impose a smoothness condition on all functions involved. Hence, the function $\boldsymbol{F}$ to be computed is assumed to be C^2, the communication protocols $\boldsymbol{f}_{ij}^k$ (i.e. the message functions), local computation including $\boldsymbol{f}^r$ by each computer and the function h are all assumed to be C^2.

Definition. An equilibrium model $(\phi_1,\ldots,\phi_s;\mathbb{R}^n;h)$ is said to be *efficient* if the Jacobians $\nabla_{x_i}\phi_i$, $i=1,\ldots,s$, are all of maximal rank. A multistage process is said to be *efficient* if the equilibrium model that it converts to is efficient.

An efficient multistage process is free of redundancy in information exchange, hence, is able to achieve the minimum communication complexity. The following proposition shows that an equilibrium model that is not efficient does not achieve minimality in the dimension of the message space in computing $\boldsymbol{F}$.

Proposition 3.2. *If* $\boldsymbol{F}$ *can be computed by an equilibrium model which is not efficient with a message space of dimension* n*, then* $\boldsymbol{F}$ *can be computed by an efficient equilibrium model with a message space of dimension less than* n.

Proof. Suppose $(\phi_1,\dots,\phi_s;\mathbb{R}^n;h)$ is the equilibrium model that computes $\boldsymbol{F}$ and is not efficient. Let $\mu(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)$ be the solution for μ in (3.1). Differentiating (3.1) yields

$$\left(\frac{\partial\mu(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)}{\partial(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)}\right)_{n\times\sum_{i=1}^s k_i}=-\left(\frac{\partial(\phi_1,\dots,\phi_s)}{\partial\mu}\right)^{-1}_{n\times n}\left(\frac{\partial(\phi_1,\dots,\phi_s)}{\partial(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)}\right)_{n\times\sum_{i=1}^s k_i}.$$

So $\partial\mu(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)/\partial(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)$ is not of maximal rank since $\partial(\phi_1,\dots,\phi_s)/\partial(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)$ is not. Hence, the component functions $\mu_1(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s),\dots,\mu_n(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)$ of $\mu(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)$ are functionally dependent. Suppose that their rank is $r<n$. Without loss of generality, we can assume that

$$\mu_i(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)=\gamma_i(\mu_1(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s),\dots,\mu_r(\boldsymbol{x}_1,\dots,\boldsymbol{x}_s)),\quad i=r+1,\dots,n,$$

for some functions $\gamma_{r+1},\dots,\gamma_n$. Let $\tilde{h}\colon\mathbb{R}^r\to\mathbb{R}^m$ and $\psi_1,\dots,\psi_s$ be defined as

$$\tilde{h}(\mu_1,\dots,\mu_r)=h(\mu_1,\dots,\mu_r,\gamma_{r+1}(\mu_1,\dots,\mu_r),\dots,\gamma_n(\mu_1,\dots,\mu_r)),$$

$$\psi_i(\boldsymbol{x}_i;\mu_1,\dots,\mu_r)=\phi_i(\boldsymbol{x}_i;\mu_1,\dots,\mu_r,\gamma_{r+1}(\mu_1,\dots,\mu_r),\dots,\gamma_n(\mu_1,\dots,\mu_r)),$$

$i=1,\dots,s$, then

$$\frac{\partial(\psi_1,\dots,\psi_s)}{\partial(\mu_1,\dots,\mu_r)}=\frac{\partial(\phi_1,\dots,\phi_s)}{\partial(\mu_1,\dots,\mu_r)}+\frac{\partial(\phi_1,\dots,\phi_s)}{\partial(\mu_{r+1},\dots,\mu_n)}\cdot\frac{\partial(\gamma_{r+1},\dots,\gamma_n)}{\partial(\mu_1,\dots,\mu_r)}.$$

So $\partial(\psi_1,\dots,\psi_s)/\partial(\mu_1,\dots,\mu_r)$ is of maximal rank since $\partial(\phi_1,\dots,\phi_s)/\partial(\mu_1,\dots,\mu_n)$ is. Therefore, we can delete some components in each of $\psi_1,\dots,\psi_s$ to get $\tilde{\phi}_1,\dots,\tilde{\phi}_s$ such that the $r\times r$ square matrix $\partial(\tilde{\phi}_1,\dots,\tilde{\phi}_s)/\partial(\mu_1,\dots,\mu_r)$ is of maximal rank. It is easily checked that the equilibrium model $(\tilde{\phi}_1,\dots,\tilde{\phi}_s;\mathbb{R}^r;\tilde{h})$ is efficient and computes $\boldsymbol{F}$. □

For an equilibrium model to compute $\boldsymbol{F}\colon\mathbb{R}^{k_1}\times\cdots\times\mathbb{R}^{k_s}\to\mathbb{R}^m$, the message space $\mathbb{R}^n$ has to be large enough, i.e. the dimension n has to be large enough. In [3, 4], the following theorem is proved, which gives a necessary condition on the dimension of the message space of an efficient equilibrium model that computes $\boldsymbol{F}=(F^1,\dots,F^m)$.

Theorem 3.1. *Let $\boldsymbol{F}\colon\mathbb{R}^{k_1}\times\cdots\times\mathbb{R}^{k_s}\to\mathbb{R}^m$ be C^2 and $U\subseteq\mathbb{R}^{k_1}\times\cdots\times\mathbb{R}^{k_s}$ be open. If $\boldsymbol{F}$ can be computed in U by an efficient equilibrium model with a message space of dimension n, then the following condition must be satisfied in U:*

$$v_1\wedge\cdots\wedge v_{n+1}=0,\quad\forall v_1,\dots,v_{n+1}\in\{Q_i^k,\,\mathrm{d}Q_i^k\mid i=1,\dots,s;\ k=1,\dots,m\},$$

where

$$Q_i^k=\sum_{j=1}^{k_i}\frac{\partial F^k}{\partial x_j^{(i)}}\,\mathrm{d}x_j^{(i)},\quad i=1,\dots,s;\ k=1,\dots,m,$$

are differential one-forms, $\mathrm{d}Q_i^k$ is the differential two-form obtained by applying the differential operator d *on Q_i^k, and $v_1\wedge\cdots\wedge v_{n+1}$ is the wedge product of the differential forms $v_1,\dots,v_{n+1}$. Also $x_j^{(i)}$ denotes the jth component of $\boldsymbol{x}_i$. (A treatment of differential forms, the differential operator, and the wedge product can be found in any standard textbook on differential geometry, for example* [9].*) Note that the differential forms $v_1,\dots,v_{n+1}$ in the theorem need not be distinct.*

Theorem 3.1 implicitly gives a lower bound for the dimension of the message space of efficient equilibrium models that compute $\boldsymbol{F}$.

Corollary 3.1. *Let $\boldsymbol{F}$ and U be as in Theorem* 3.1. *If $\boldsymbol{F}$ can be computed in U by an efficient equilibrium model with a message space of dimension n, then*

$$n \geqslant \min\{t \mid v_1 \wedge \cdots \wedge v_{t+1} = 0,\ \forall v_1, \ldots, v_{t+1} \in W\}$$

or equivalently

$$n \geqslant \max\{t \mid v_1 \wedge \cdots \wedge v_t \neq 0, \text{ for some } v_1, \ldots, v_t \in W\},$$

where $W = \{Q_i^k, \mathrm{d}Q_i^k \mid i = 1, \ldots, s;\ k = 1, \ldots, m\}$.

Combining Corollary 3.1 with Propositions 3.1 and 3.2, we obtain the following lower bound on communication complexity.

Theorem 3.2. *Let $\boldsymbol{F}: \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_s} \to \mathbb{R}^m$ be C^2 and $U \subseteq \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_s}$ be open. If $\boldsymbol{F}$ can be computed in U by a multistage process with communication complexity N, then*

$$N \geqslant \min\{t \mid v_1 \wedge \cdots \wedge v_{t+1} = 0,\ \forall v_1, \ldots, v_{t+1} \in W\} - m$$

or equivalently

$$N \geqslant \max\{t \mid v_1 \wedge \cdots \wedge v_t \neq 0, \text{ for some } v_1, \ldots, v_t \in W\} - m,$$

where $W = \{Q_i^k, \mathrm{d}Q_i^k \mid i = 1, \ldots, s;\ k = 1, \ldots, m\}$ *and*

$$Q_i^k = \sum_{j=1}^{k_i} \frac{\partial F^k}{\partial x_j^{(i)}} \mathrm{d}x_j^{(i)}.$$

Once again note that the differential forms $v_1, \ldots, v_{t+1}$ in the theorem need not be distinct.

In the case where $s = 2$ and $m = 1$, namely, a scalar function $F(\boldsymbol{x}, \boldsymbol{y})$ is to be computed by a two-computer network, the set in the theorem $W = \{Q_x, Q_y, \mathrm{d}Q_x, \mathrm{d}Q_y\}$, where

$$Q_x = \sum_{i=1}^{k_1} \frac{\partial F}{\partial x_i} \mathrm{d}x_i, \quad Q_y = \sum_{j=1}^{k_2} \frac{\partial F}{\partial y_j} \mathrm{d}y_j, \quad \mathrm{d}Q_y = -\mathrm{d}Q_x = \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} \frac{\partial^2 F}{\partial x_i \partial y_j} \mathrm{d}x_i \mathrm{d}y_j.$$

The lower bound is given by $\max\{t \mid v_1 \wedge \cdots \wedge v_t \neq 0, \text{ for some } v_1, \ldots, v_t \in W\} - 1$, which can be shown to be equal to the rank of the following matrix minus one:

$$BH(F) = \begin{pmatrix} F_{x_1 y_1} & F_{x_1 y_2} & \cdots & F_{x_1 y_{k_2}} & F_{x_1} \\ F_{x_2 y_1} & F_{x_2 y_2} & \cdots & F_{x_2 y_{k_2}} & F_{x_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ F_{x_{k_1} y_1} & F_{x_{k_1} y_2} & \cdots & F_{x_{k_1} y_{k_2}} & F_{x_{k_1}} \\ F_{y_1} & F_{y_2} & \cdots & F_{y_{k_2}} & 0 \end{pmatrix}_{(k_1+1)\times(k_2+1)}$$

where

$$F_{x_iy_j}=\frac{\partial^2 F}{\partial x_i\,\partial y_j},\quad F_{x_i}=\frac{\partial F}{\partial x_i},\qquad F_{y_j}=\frac{\partial F}{\partial y_j},\qquad i=1,\ldots,k_1;\ j=1,\ldots,k_2.$$

So we have the following corollary.

Corollary 3.2. *Let* $F:\mathbb{R}^{k_1}\times\mathbb{R}^{k_2}\to\mathbb{R}$ *be* C^2 *and* $U\subseteq\mathbb{R}^{k_1}\times\mathbb{R}^{k_2}$ *be open. If* F *can be computed in* U *by a multistage process with communication complexity* N, *then* $N\geqslant \operatorname{rank} BH(F)-1$.

Note that in this special case our lower bound is similar to that of Abelson [1]. Indeed, they can differ by at most 1.

4. Computability of the lower bound and some applications

From Theorem 3.2, we see that the lower bound for the communication complexity of computing $\boldsymbol{F}$ is the smallest of all numbers t minus m such that any $t+1$ (not necessarily distinct) differential forms from the set W have wedge product zero, or equivalently, is the largest of all numbers t minus m such that there are t (not necessarily distinct) differential forms from the set W whose wedge product is not zero. This lower bound has the desirable feature that it is practically computable, because computation of differential forms, the differential operation, and the wedge product can be easily carried out either numerically or symbolically on the computer. A primitive algorithm for computing the lower bound is as follows.

Algorithm for lower bound computation

(1) input function $\boldsymbol{F}$;
(2) compute set W;
(3) let $W^*=W$;
(4) let lower bound $=0$;
(5) while W^* is not empty do
    let W^* be the set of all $w_1\wedge w_2$ which are not zero, where $w_1\in W^*$ and $w_2\in W$;
    lower bound = lower bound + 1
  endwhile;
(6) lower bound = lower bound $-$ m;
(7) output lower bound.

It is also worth pointing out that the computation of the lower bound is highly parallelizable and can be implemented by parallel or distributed algorithms.

As an illustration, we now use Theorem 3.2 to answer the question posed in Section 2. We have seen that function $F:\mathbb{R}^{k_1}\times\mathbb{R}^{k_2}\times\mathbb{R}^{k_3}\to\mathbb{R}$, defined by

$$F(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})=\sum_{i=1}^{k_1}x_i(p_2(y_1))^i+\sum_{i=1}^{k_2}y_i(p_3(z_1))^i+\sum_{i=1}^{k_3}z_i(p_1(x_1))^i,$$

can be computed by a multistage process with exchange of 5 messages. By applying Theorem 3.2 we can show that no multistage process can compute this function with exchange of fewer than 5 messages. To simplify the calculation, assume that the functions p_1, p_2, p_3 are all the identity function. It can be calculated that the set in Theorem 3.2 $W=\{Q_x, Q_y, Q_z, \mathrm{d}Q_x, \mathrm{d}Q_y, \mathrm{d}Q_z\}$, where

$$Q_x=\left(y_1+\sum_{i=1}^{k_3} iz_i x_1^{i-1}\right)\mathrm{d}x_1+\sum_{i=2}^{k_1} y_1^i\,\mathrm{d}x_i,$$

$$Q_y=\left(z_1+\sum_{i=1}^{k_1} ix_i y_1^{i-1}\right)\mathrm{d}y_1+\sum_{i=2}^{k_2} z_1^i\,\mathrm{d}y_i,$$

$$Q_z=\left(x_1+\sum_{i=1}^{k_2} iy_i z_1^{i-1}\right)\mathrm{d}z_1+\sum_{i=2}^{k_3} x_1^i\,\mathrm{d}z_i,$$

$$\mathrm{d}Q_x=\left(\sum_{i=1}^{k_3} ix_1^{i-1}\,\mathrm{d}z_i\right)\wedge\mathrm{d}x_1-\left(\sum_{i=1}^{k_1} iy_1^{i-1}\,\mathrm{d}x_i\right)\wedge\mathrm{d}y_1,$$

$$\mathrm{d}Q_y=\left(\sum_{i=1}^{k_1} iy_1^{i-1}\,\mathrm{d}x_i\right)\wedge\mathrm{d}y_1-\left(\sum_{i=1}^{k_2} iz_1^{i-1}\,\mathrm{d}y_i\right)\wedge\mathrm{d}z_1,$$

$$\mathrm{d}Q_z=\left(\sum_{i=1}^{k_2} iz_1^{i-1}\,\mathrm{d}y_i\right)\wedge\mathrm{d}z_1-\left(\sum_{i=1}^{k_3} ix_1^{i-1}\,\mathrm{d}z_i\right)\wedge\mathrm{d}x_1.$$

It can be verified that

$$\begin{aligned}&Q_x\wedge Q_y\wedge Q_z\wedge\mathrm{d}Q_x\wedge\mathrm{d}Q_x\wedge\mathrm{d}Q_y\\&\quad=2\left(\sum_{i<j}(j-i)y_1^{i+j-1}\,\mathrm{d}x_i\,\mathrm{d}x_j\right)\wedge\left(\sum_{i<j}(j-i)z_1^{i+j-1}\,\mathrm{d}y_i\,\mathrm{d}y_j\right)\\&\qquad\wedge\left(\sum_{i<j}(j-i)x_1^{i+j-1}\,\mathrm{d}z_i\,\mathrm{d}z_j\right)\wedge\mathrm{d}x_1\,\mathrm{d}y_1\,\mathrm{d}z_1\neq 0.\end{aligned}$$

So by Theorem 3.2, the lower bound is 5. Any multistage process that computes this function needs to transmit at least 5 real values between the computers. The three-stage process described in Section 2 is optimal in that it requires the minimum intercomputer communication.

As another illustration, consider computing the determinant of an $n\times n$ matrix $A=(a_{ij})_{n\times n}$ in an n-computer network under the assumption that computer CX_i has access to column i only, $i=1,\ldots,n$. It is easy to see that $Q_i=\sum_{k=1}^{n} A_{ki}\,\mathrm{d}a_{ki}$, where A_{ki} is the algebraic cofactor of a_{ki}. It can be calculated that for $A=I_{n\times n}$, the $n\times n$ unit matrix, $Q_i=\mathrm{d}a_{ii}$, $\mathrm{d}Q_i=\sum_{k=1}^{n}(\mathrm{d}a_{kk}\,\mathrm{d}a_{ii}-\mathrm{d}a_{ik}\,\mathrm{d}a_{ki})$, $i=1,\ldots,n$. It can further be verified that

$$\left(\bigwedge_{i=1}^{n} Q_i\right)\wedge\left(\bigwedge_{i=1}^{n-1}(\mathrm{d}Q_i)^i\right)=(-1)^{(n-1)n/2}\left(\prod_{k=1}^{n-1} k!\right)\mathrm{d}a_{11}\cdots\mathrm{d}a_{nn}\neq 0.$$

So we obtain the following lower bound from Theorem 3.2.

Corollary 4.1. *Computing the determinant of an $n \times n$ matrix in a network of n computers as described above has a communication complexity of at least $n(n+1)/2-1$.*

As a final illustration, consider solving a system of n linear equations in n unknowns, $AX=b$, in an n-computer network under the assumption that different columns of the coefficient matrix A are accessed by different computers and b is accessed by all computers. This is equivalent to computing $X=A^{-1}b$ (assuming that A is invertible). It can be calculated that for $A=I_{n\times n}$, the $n\times n$ unit matrix, $Q_i^l=-b_i\,\mathrm{d}a_{li}$, $i=1,\ldots,n$; $l=1,\ldots,n$. So

$$\bigwedge_{l=1}^{n}\bigwedge_{i=1}^{n} Q_i^l=(-1)^{n^2}\left(\prod_{i=1}^{n} b_i^n\right)\mathrm{d}a_{11}\cdots\mathrm{d}a_{nn}\neq 0,$$

if all $b_i\neq 0$. So from Theorem 3.2 we obtain the lower bound n^2-n, which is the obvious upper bound.

Corollary 4.2. *Solving a system of n linear equations in n unknowns in a network of n computers in the way described above has communication complexity n^2-n.*

Abelson [1] also considered computing the determinant of an $n\times n$ matrix and solving a system of linear equations in such a setting. He applied his lower bound for the two-computer case to the n-computer network by dividing the whole network into two halves in all possible ways and obtained a lower bound of $n^2/2$ for the problem of computing the $n\times n$ determinant and a lower bound of $n(n-1)/2$ for the problem of solving a linear system. So the lower bound we obtain here for the problem of computing the determinant provides some improvement, while the lower bound we obtain here for the problem of solving a linear system is the best possible.

The problems of computing the determinant and of solving the linear system are additional examples of computation problems for which the communication cost is a significant part of the total computation cost. It is known (see e.g. [2, Section 2.2]) that without considering the communication cost the complexity of typical algorithms for the two problems using n computers is $O(n^2)$, to which the communication cost is comparable.

5. Concluding remarks

Our lower bound result is a very general one. It can be used to obtain a lower bound on the communication complexity for any distributed computation problem as long as the problem can be formulated as the problem of evaluating a differentiable function. This class of problems includes many that are of theoretical significance as well as many that are found in practical numerical computation. For example, many matrix computation problems are of this kind.

On the other hand, the lower bound so obtained is not always tight. It is easy to see that the best possible value that can be obtained from the bound for computing

a function $F: \mathbb{R}^n \times \cdots \times \mathbb{R}^n$ (n times)$\to \mathbb{R}$ is $n(n+1)/2-1$, which is unlikely to be the absolute upper bound. It seems that there exists a function that has a communication complexity as high as n^2-n, which is the obvious upper bound and can be achieved by having one computer receive all the data from the other computers and all the computation done there. If that is the case, then the lower bound still leaves much room for improvement.

Also the lower bound may perform very poorly for some problems, for example, computing one single entry of the inverse matrix $(X+Y)^{-1}$ with two computers, where X and Y are two $n \times n$ matrices (see [6]), or solving a polynomial equation of the form $\sum_{i=0}^{n-1}(x_i+y_i)z^i=0$ with two computers (see [7]). It is, therefore, desirable to understand for what kind of problems the bound performs well and for what kind of problems it does not.

Finally, we mention the question that is still left open by our lower bound result, namely, to determine the communication complexity of computing an $n \times n$ determinant.

Acknowledgment

I would like to thank D. Saari and G. Wasilkowski for comments on an earlier draft of this paper and S. Reiter for bringing [1] to my attention. The results in this paper are based, in part, on my Ph.D. Thesis at Northwestern University. Partial support from NSF grant IRI-8803505 is also gratefully acknowledged. The paper was presented at the Sixth SIAM Conference on Discrete Mathematics (Vancouver, Canada, June 1992) and at the First International Symposium of Young Investigators on Information/Computer/Control (Beijing, China, February 1994). An extended abstract appeared in the proceedings of the latter.

References

[1] H. Abelson, Lower bounds on information transfer in distributed computations, *J. Assoc. Comput. Mach.* **27** (1980) 384–392.

[2] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[3] P. Chen, On the efficiency and complexity of computational and economic processes, Ph.D. Thesis, Northwestern University, 1989.

[4] P. Chen, A lower bound for the dimension of the message space of the decentralized mechanisms realizing a given goal, *J. Math. Econom.* **21** (1992) 249–270.

[5] W.M. Gentleman, Some complexity results for matrix computations on parallel processors, *J. Assoc. Comput. Mach.* **25** (1978) 112–115.

[6] Z.-Q. Luo and J.N. Tsitsiklis, On the communication complexity of distributed algebraic computation, preprint (1989).

[7] Z.-Q. Luo and J.N. Tsitsiklis, On the communication complexity of solving a polynomial equation, *SIAM J. Comput.* **20** (1991) 936–950.

[8] K. Mount and S. Reiter, The informational size of message spaces, *J. Econom. Theory* **8** (1974) 161–192.

[9] M. Spivak, *A Comprehensive Introduction to Differential Geometry, Vol. I* (Publish or Perish, Berkeley, CA, 1979).